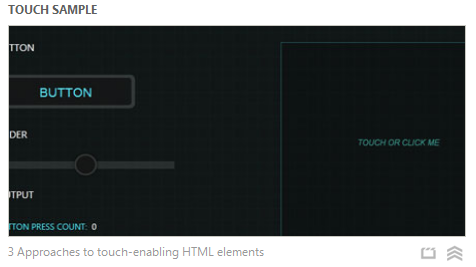
|  |
| --- |
| HTML5 Elements Cookbook |
| H5E Experiment: Touch Events |
| DRAFT  Published 15 July, 2011 |
|  |

|  |
| --- |
|  |
| H5E Scout Team  Windows Web Partners  Microsoft Corporation  Microsoft Confidential |

# Summary

On devices where the user interacts directly with the screen such as a phone or tablet, the browser sometimes provides dedicated events to indicate that the user is touching the screen. These *touch events* allow finer control over the timing and interaction on the screen. By using touch events when they are available, an HTML application can provide a better and more responsive experience.



## Scope

This document describes the use of browser touch events. This document assumes an existing knowledge of JavaScript and jQuery. This document does not supersede any requirements or instructions provided by the IE team.

## Keywords

Touch events, touchstart, touchend, touchmove, mouse events, mouseover, mouseout, mousemove, click, jQuery

## Contact us

To contact us for questions or support, please email Chewy Chong ([ChewyC](mailto:ChewyC?subject=HTML5%20Cookbooks)). Feedback is welcome.

Contents

Summary 2

Scope 2

Keywords 2

Contact us 2

Inspiration for this experiment 4

Walkthrough 4

Details 6

How do I build this using HTML5? 7

Using jQuery.ui.touch 7

Utilizing direct support for touch events 8

Implementing with jQuery Mobile 9

Conclusions and Recommendations 10

Resources 10

Relevant Web sites and specifications 10

Microsoft Resources 10

Appendix A: About H5E 11

What is an HTML5 Elements Cookbook? 11

Contact us 11

Copyright and trademark information 11

Document Revision History 12

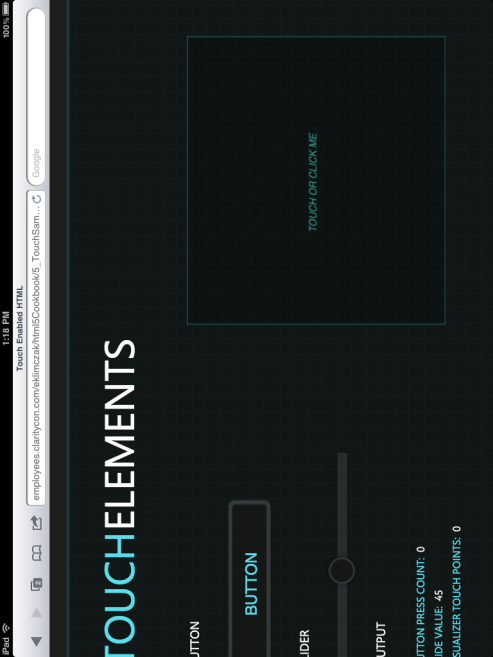
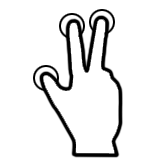
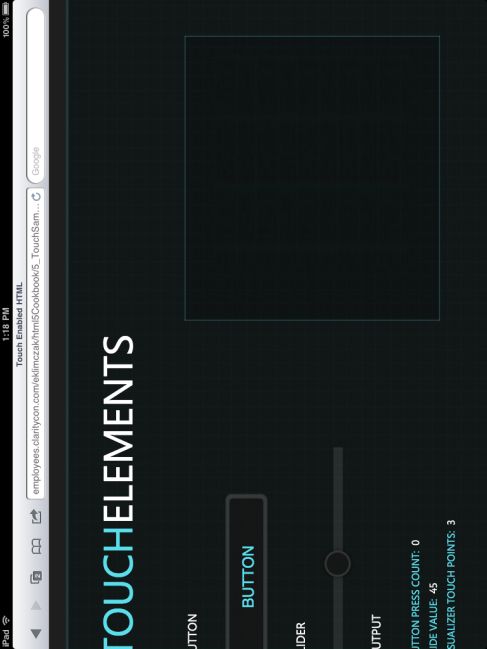
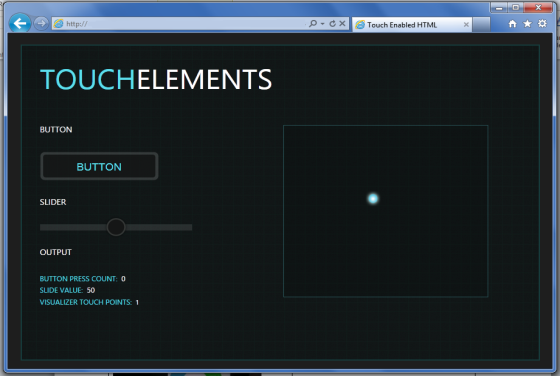
# Inspiration for this experiment

Several mobile platforms such as the iPhone, iPad, and Android have special browser “touch” events to indicate when the user is interacting with the screen. These events provide much better feedback than mouse events on the platforms where they are implemented. For example, the click event on a link in a web page delays about one-half a second, but the **touchstart** and **touchend** events on the link are nearly instantaneous. Also, mobile browsers do not always deliver click events to all elements on the page, only “clickable” elements such as links or buttons. HTML applications and web pages that are not touch-aware may appear sluggish or work incorrectly because they wait for the click event to be delivered.

The simplest way for existing HTML applications and pages to support touch events is to add a shim that translates touch events into mouse events where possible. This allows the code to take advantage of the faster response time of touch events without requiring significant changes in code or page design. That is the approach taken by the jQuery.ui.touch plugin used by some components of this experiment. A different approach must be taken for situations where an application wants to support multiple touches on a screen. There, it is best to feature-detect the touch event capability and directly handle touch events.

## Walkthrough

The Touch Events sample is designed to target an iPad. The demo application includes the following behavior:

1. The user navigates to the page, and the default screen appears.  
   
2. Touch demonstration page on an iPad, with three fingers touching the canvas.  
   
3. With IE9 on a desktop, the mouse acts as a single touch point when clicked.  
   

## Details

Touch events are not currently defined by any standard, and their behavior can be quirky across different platforms (<http://blogs.adobe.com/adobeandjquery/2011/03/07/the-current-state-of-touch-events/>). The W3C is in the early stages of standardization for touch events (<http://www.w3.org/TR/2011/WD-touch-events-20110505/>) but it will likely be several years before implementations converge on consistent behavior. At present, the draft standard allows significant leeway in how mouse events are delivered on a touch-capable platform, so even when the standard is widely implemented there could be inconsistencies.

# How do I build this using HTML5?

***NOTE****: Many of the HTML5 Experiments are still under development. Our initial target is to build prototypes that work on current HTML5-supported browsers and tablet devices, including the iPad. The experiments do not aim for full cross-browser support at this stage, but we will likely build in graceful degradation in future updates.*

To ensure that users have a similar cross-browser experience, the following table describes the compatibility of the solutions in this document:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| HTML5 Logo**HTML5 Feature** | IE6.0 | IE7.0 | IE8.0 | IE9.0 | IEPP | Chrome11 | | Chrome12 | Safari4.x | Safari5.x | Firefox3.6 | Firefox4.x | Firefox5.x | Opera11 |
| jQuery |  |  |  |  |  |  | |  |  |  |  |  |  |  |
|  | | | | | | | Full Support | | | | | | |  |
|  | | | | | | | Supported with Shim | | | | | | |  |
|  | | | | | | | No current support | | | | | | |  |

***NOTE****: In general, shims are not incorporated into the HTML5 Experiments at this stage. If a shim or polyfill is required for cross-browser support, see* [*http://browserexperiments.com*](http://browserexperiments.com) *for details on shim implementation.*

## Primary files in this solution

### Source Location

<https://github.com/molant/BrowserExperiments/tree/master/cookbook/5_TouchSample>

### Sample Location

<http://employees.claritycon.com/eklimczak/html5Cookbook/5_TouchSample/>

## Using jQuery.ui.touch

The jQuery.ui touch shim (<https://github.com/molant/BrowserExperiments/blob/master/cookbook/5_TouchSample/js/jquery.ui.touch.js>) allows pages to use the standard mouse events such as click or mouseover on platforms that support touch events. It only requires the core jQuery library; although its name may imply it requires jQuery UI to work properly, it does not. It is, however, often used with jQuery UI widgets to provide touch support for them. This demonstration shows how it can be used with a jQuery UI slider control so that touch events can move the slider. The slider is created using the standard jQuery UI slider setup:

//Create Slider with jqueryUI

$(".slider").slider({

animate: true,

range: "min",

value: 50,

min: 0,

max: 100,

step: 1,

slide: function (event, ui) {

$("#slideResult").html("SLIDE VALUE:&nbsp;&nbsp;" + "<span style='color:#fff;'>" +ui.value + "</span>");

}

});

Then the jQuery.ui.touch code is used to convert touch events to the mouse events the UI slider requires:

//handlers in jquery.ui.touch.js

addTouch($('.slider')[0]);

function addTouch(el){

el.addEventListener("touchstart", TouchHandler, false);

el.addEventListener("touchmove", TouchHandler, false);

el.addEventListener("touchend", TouchHandler, false);

el.addEventListener("touchcancel", TouchHandler, false);

}

The **TouchHandler** function in jQuery.ui.touch.js examines the touch events as they come in; depending on the order and timing of the touch events it generates corresponding mouse events.

* **Pros**: Allows touch events to be easily retrofitted into current code, since it simulates the standard click and mouse events the code already expects.
* **Cons**: Currently does not support IE6, IE7, IE8, or IE7 Mobile, which do not have **addEventListener**.

## Utilizing direct support for touch events

Simulated mouse events may not be sufficient for applications that need more detail about the user’s interaction with the screen. This is particularly true for applications that wish to support multi-touch, where multiple fingers may be touching and/or moving on the screen at once. In those cases it may be necessary to detect the presence of touch events on the platform and provide some sort of fallback in the application if touch is not supported.

Modernizr provides the ability to detect support for touch events, or several simple code snippets can instead be used to detect touch events. jQuery.ui.touch creates a $.support.touch Boolean value that can be tested to see if the platform has touch events. The larger challenge in many designs will be to determine what experience to give users on platforms that do not support touch. For example, the feature could be hidden in the application, or degraded somehow to support only the single touch point that is obtainable using a mouse. The latter is the approach used for the Canvas Touch Visualizer (<https://github.com/molant/BrowserExperiments/blob/master/cookbook/5_TouchSample/js/TouchVisualizer.js>) used in the demonstration page. The code has different paths for touch and mouse support:

if (this.touchEnabled) {

this.canvas.addEventListener('touchstart', this.OnTouchDown, false);

this.canvas.addEventListener('touchmove', this.OnTouchMove, false);

this.canvas.addEventListener('touchend', this.OnTouchEnd, false);

this.canvas.addEventListener("touchcancel", this.OnTouchEnd, false);

} else {

$(this.canvas).mousedown(this.OnMouseDown);

$(this.canvas).mouseup(this.OnMouseUp);

$(this.canvas).mousemove(this.OnMouseMove);

}

When only a mouse is supported, clicking on the canvas will show the one point tracked by the mouse position. On a touch-capable platform, it is possible to put multiple fingers on the canvas and move them at once.

* **Pros**: Supports all touch features, including multi-touch, to the best capability of the platform.
* **Cons**: Must feature-detect the presence of touch events and give a different experience for platforms that are not touch enabled. In the worst case, there may be two completely independent code paths.

## Implementing with jQuery Mobile

Although not shown in this experiment, there is an additional alternative. The jQuery Mobile platform (<http://jquerymobile.com/>) provides a compatibility layer that delivers a consistent set of events across all its supported platforms, regardless of whether the platform has touch events. It does this by introducing a new set of “virtual” mouse events (<http://jquerymobile.com/blog/2011/03/31/jquery-mobile-alpha-4-released/>). Application code uses the virtual mouse events rather than directly using touch events. On platforms that do not support touch events, the virtual mouse events are simply the standard mouse events (click, mousedown, mousemove, etc.) provided by the browser. For platforms with touch events, jQuery Mobile translates them into virtual mouse events in a consistent way.

* **Pros**: No special code paths required to allow touch events to enhance mouse behavior across all platforms, including all versions of Internet Explorer.
* **Cons**: Requires code to be written using a new set of virtual events rather than standard mouse events. Virtual events do not support multi-touch scenarios, which must be handled as in the experiment above. Also adds a dependency and requires a learning curve for a new framework; jQuery Mobile is relatively popular and stable, but still in beta so it may change quickly.

# Conclusions and Recommendations

By far, the simplest and most straightforward way to take advantage of touch events is to use a shim such as jQuery.ui.touch that converts touch events to mouse events. This provides the responsiveness of touch events without any significant code rewrite, and allows very good compatibility with non-touch platforms such as the desktop. For basic emulation of clicks and mouse movements, most applications will find this to be a solution that does not introduce significant compromises.

In situations where an application developer wants to leverage touch-specific functionality such as multi-touch, it can be feature-detected using a library such as Modernizr. If present, the application can directly attach to touch events and implement platform-specific code. For non-touch environments it may be possible to implement a mouse-oriented version with reduced functionality, as shown in the Canvas part of this experiment.

For projects are specifically targeted to mobile platforms, jQuery Mobile offers a set of virtual mouse events that automatically take advantage of touch events when they are available. This requires the code to use the virtual events (e.g., vclick instead of click) so it is more suitable to new development than to retrofitting existing code. Potentially it could be combined with targeted use of jQuery.ui. touch to allow older plugins to work properly.

# Resources

## Relevant Web sites and specifications

|  |  |
| --- | --- |
| W3C draft for touch events | <http://www.w3.org/TR/2011/WD-touch-events-20110505/> |
| jQuery Mobile | <http://jquerymobile.com/> |

## Microsoft Resources

|  |  |
| --- | --- |
| H5E Primary Contact | Chewy Chong ([ChewyC](mailto:%20chewyc?subject=HTML5%20Experiments%20Cookbooks)) |
| H5E Development Contact | Anton Molleda Quintana ([v-anmoll](mailto:v-anmoll?subject=HTML5%20Experiments%20Cookbooks)) |

# Appendix A: About H5E

## What is an HTML5 Elements Cookbook?

Each HTML5 Elements Cookbook reflects a case study of an aspirational experience that is provided by a native or component-based application. The HTML5 Experiments that are conducted by the H5E team use HTML5 and related technologies to replicate these experiences. Our primary objective is to learn from these experiments to determine if an HTML5 alternative to component-based or native implementations is both possible, and practical. Each Cookbook provides a description of the element and technical details of the HTML5 replication of that feature. We also include recommendations on whether it makes sense to pursue this approach.

## Contact us

If you need assistance with technical solutions or have a best practice to share, please contact us by sending email to Chewy Chong ([ChewyC](mailto:chewyc?subject=H5E%20Cookbooks%20and%20Documentation)).

## Copyright and trademark information

This document is Microsoft Confidential.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

 2011 Microsoft Corporation. All rights reserved.

# Document Revision History

|  |  |  |
| --- | --- | --- |
| **Reviser** | **Date** | **Revisions** |
| **v-davime** | 14 July 2011 | Initial draft |
| **v-jgeige** | 18 July 2011 | Minor edits |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |